NPS-53-89-002

NAVAL POSTGRADUATE SCHOOL

Monterey, California

FORTRAN SUBROUTINES FOR UPDATING
THE QR DECOMPOSITION

William Gragg
Lother Reichel

November 1988

Approved for public release; distribution unlimited
Prepared for: Naval Postgraduate School and the
National Science Foundation, Washington
D.C. 20550

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-53-89-002	5 MONITORING ORGANIZATION REPORT NUMBER(S) NPS-53-89-002
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL (If applicable) 53
7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School and the National Science Foundation	
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943 and Washington, D.C.
8a NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School	8b OFFICE SYMBOL (If applicable) 53
9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER O&MN, Direct funding	
8c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO. / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

FORTRAN Subroutines For Updating The QR Decomposition

12 PERSONAL AUTHOR(S)
William Gragg and L. Reichel

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical Report	FROM 1 Oct '87 TO 1 Oct '88	10 November 1988	21

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	QR decomposition, updating, subset selection

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Abstract: Let the matrix $A \in \mathbf{R}^{m \times n}$, $m \geq n$, have a QR decomposition $A = QR$, where $Q \in \mathbf{R}^{m \times n}$ has orthonormal columns, and $R \in \mathbf{R}^{n \times n}$ is upper triangular. Assume that Q and R are explicitly known. We present FORTRAN subroutines that update the QR decomposition in a numerically stable manner when A is modified by a matrix of rank one, or when a row or a column is inserted or deleted. These subroutines are modifications of the Algol procedures in Daniel et al. [5]. We also present a subroutine that permutes the columns of A and updates the QR decomposition so that the elements in the lower right corner of R will generally be small if the columns of A are nearly linearly dependent. This subroutine is an implementation of the rank revealing QR decomposition scheme recently proposed by Chan [3]. The subroutines have been written to perform well on a vector computer.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL William Gragg	22b TELEPHONE (Include Area Code) (408) 646-2194 / 22c OFFICE SYMBOL 53Gr

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

FORTRAN Subroutines for Updating the QR Decomposition*

L. Reichel
Bergen Scientific Centre
and
University of Kentucky

W.B. Gragg
Naval Postgraduate School
and
University of Kentucky

Abstract: Let the matrix $A \in \mathbf{R}^{m \times n}$, $m \geq n$, have a QR decomposition $A = QR$, where $Q \in \mathbf{R}^{m \times n}$ has orthonormal columns, and $R \in \mathbf{R}^{n \times n}$ is upper triangular. Assume that Q and R are explicitly known. We present FORTRAN subroutines that update the QR decomposition in a numerically stable manner when A is modified by a matrix of rank one, or when a row or a column is inserted or deleted. These subroutines are modifications of the Algol procedures in Daniel et al. [5]. We also present a subroutine that permutes the columns of A and updates the QR decomposition so that the elements in the lower right corner of R will generally be small if the columns of A are nearly linearly dependent. This subroutine is an implementation of the rank revealing QR decomposition scheme recently proposed by Chan [3]. The subroutines have been written to perform well on a vector computer.

Categories and subject descriptors: G.1.3 [Numerical Analysis]: Numerical Linear Algebra; G.4 [Mathematics of Computing]: Mathematical Software

General Terms: Algorithms
Additional Key Words and Phrases: QR decomposition, updating, subset selection.

Authors' addresses: L. Reichel, Bergen Scientific Centre, Allegaten 36, N-5007 Bergen, Norway; and University of Kentucky, Department of Mathematics, Lexington, KY 40506, USA; W.B. Gragg, Naval Postgraduate School, Department of Mathematics, Monterey, CA 93943, USA.

*Research supported by the National Science Foundation under Grant DMS-870416 and by the Naval Postgraduate School Research Council

1. INTRODUCTION

The purpose of this paper is to present several FORTRAN subroutines for updating the QR decomposition of a matrix. Let $A \in \mathbf{R}^{m\times n}$, $m \geq n$, have a QR decomposition $A = QR$, where $Q \in \mathbf{R}^{m\times n}$ has orthonormal columns, and $R \in \mathbf{R}^{n\times n}$ is upper triangular. Assume that the elements of Q and R are explicitly known. Let $\bar{A} \in \mathbf{R}^{p\times q}$, $p \geq q$, be obtained from A by inserting or deleting a row or a column, or let $\bar{A}$ be a rank-one modification of A, i.e., $\bar{A} = A + vu^T$, where $u \in \mathbf{R}^n$, $v \in \mathbf{R}^m$. Then a QR-decomposition of $\bar{A}$, $\bar{A} = \bar{Q}\bar{R}$, where $\bar{Q} \in \mathbf{R}^{p\times q}$ has orthonormal columns and $\bar{R} \in \mathbf{R}^{q\times q}$ is upper triangular, can be computed in $O(mn)$ arithmetic operations by updating Q and R; see Daniel et al. [5]. The updating is done by applying Givens reflectors. The operation count for updating Q and R compares favorably with the $O(mn^2)$ arithmetic operations necessary to compute a QR decomposition of a general $m\times n$ matrix.

Algol procedures for computing $\bar{Q}$ and $\bar{R}$ from Q and R are presented by Daniel et al. [5]. Buckley [2] translated these procedures into FORTRAN. Our FORTRAN subroutines implement modifications of the Algol procedures in [5]. These modifications speed up the subroutines and make them suitable for use on vector computers. This is illustrated by timing experiments.

Several program libraries, such as LINPACK [6] and NAG [14], provide subroutines for updating R only, but contain no routines for updating the complete QR decomposition. Advantages of updating both Q and R include that downdating can be carried out stably, and that the individual elements of projections are easily accessible; see LINPACK [6, p. 10.23], Daniel et al. [5], and Stewart [17].

The first comprehensive survey of updating algorithms was presented by Gill et al. [8], and a recent discussion with references to applications can be found in Golub and Van Loan [10, Chapter 12.6]. The applications include linear least squares problems, regression analysis, and the solution of nonlinear systems of equations. See Allen [1], Goldfarb [9], Gragg and Stewart [11], More and Sorensen [13]. The algorithms would also appear to be applicable to recursive least squares problems of signal processing; see Ling et al. [12].

We also present a subroutine which implements the rank revealing QR decomposition method recently proposed by Chan [3]. In this method the QR decomposition $A = QR$ is updated to yield the QR decomposition $\bar{A} = \bar{Q}\bar{R}$, where $\bar{A}$ is obtained from A by column permutation. This permutation is selected so that, in general, the element(s) in the lower right corner of $\bar{R}$ are small if A has nearly linearly dependent columns. The subroutine can be used to solve the subset selection problem; see Golub and Van Loan [10]. Table 1.1 lists the FORTRAN subroutines for updating the QR decomposition. All subroutines use double precision arithmetic and are written in FORTRAN 77. Section 2 contains programming details for the subroutines of Table 1.1 and for certain auxiliary subprograms. For all subroutines of Table 1.1, except DRRPM, the numerical method as well as Algol procedures have been presented in [5]. For these subroutines we will only discuss differences between our FORTRAN subroutines and the Algol procedures. These differences stem in part from the algorithms being sped up, as well as from the use of simple subroutines, BLAS, for elementary vector and matrix operations.

Subroutine	Purpose
DDELC	Computes $\bar{Q},\bar{R}$ from Q,R when $\bar{A}$ is obtained from A by deleting a column; see [5].
DDELR	Computes $\bar{Q},\bar{R}$ from Q,R when $\bar{A}$ is obtained from A by deleting a row; see [5].
DINSC	Computes $\bar{Q},\bar{R}$ from Q,R when $\bar{A}$ is obtained from A by inserting a column; see [5].
DINSR	Computes $\bar{Q},\bar{R}$ from Q,R when $\bar{A}$ is obtained from A by inserting a row; see [5].
DRNK1	Computes $\bar{Q},\bar{R}$ from Q,R when $\bar{A}$ is a rank-one modification of A; see [5].
DRRPM	Computes $\bar{Q},\bar{R}$ from Q,R when $\bar{A}$ is obtained by permuting the columns of A in a manner that generally reveals if columns of A are nearly linearly dependent; see [3].

Table 1.1: Subroutines for updating a QR decomposition $A = QR$ to yield a QR decomposition $\bar{A} = \bar{Q}\bar{R}$.

The BLAS are discussed in Section 3. They have been written to vectorize efficiently on a IBM 3090-200VF computer using the vectorizing compiler VS FORTRAN 2.3.0 without special compiler directives. Most BLAS were obtained by modifying LINPACK BLAS [6]. We hope that the provided BLAS vectorize well without excessive timing increases also on other vector computers. Section 4 contains output from a driver illustrating the use of the subroutines. A listing of the source code of the driver is provided in the Appendix. Section 4 also contains some timing results.

2. THE UPDATING SUBROUTINES

We consider the subroutines of Table 1.1 in order. These subroutines use auxiliary subroutines which we need to introduce first. They are listed in Table 2.1.

Auxiliary subroutine	Called by subroutine	Purpose
DORTHO	DINSC, DRNK1	Compute s: = $Q^T w$, v: = $(I-QQ^T)w$ with reorthogonalization for arbitrary vector w.
DORTHX	DDELR	Compute s: = $Q^T e_j$, v: = $(I-QQ^T)e_j$, with reorthogonalization for axis vector e_j.
DINVIT	DRRPM	Compute approximation of a right singular vector corresponding to a least singular value of R. A first approximation is obtained from the LINPACK condition number estimator DTRCO, and is improved by inverse iteration.
DTRLSL	DINVIT	Solve lower triangular system of equations with frequent rescalings in order to avoid overflow. Similar to part of DTRCO.
DTRUSL	DINVIT	Solve upper triangular linear system of equations with frequent rescalings in order to avoid overflow. Similar to part of DTRCO.

Table 2.1: Auxiliary subroutines.

2.1 Subroutines DORTHO and DORTHX

Given a matrix $Q \in \mathbb{R}^{m \times n}$, $m \geq n$, with orthonormal columns and a vector $w \in \mathbb{R}^m$, the subroutine DORTHO computes the Fourier coefficients s: = $Q^T w$ and the orthogonal projection of w into the null-space of Q^T, v: = $(I-QQ^T)w$. At most one reorthogonalization is

carried out. Since the subroutine DORTHO differs from the corresponding Algol procedure "orthogonalize" in [5] we discuss DORTHO and its use in some detail.

Subroutine DORTHO is called by routine DINSC, which updates the QR factorization of a matrix $A = QR \in \mathbf{R}^{m \times n}$, $m > n$, when a column w is inserted into A. Updating may not be meaningful if w is nearly a linear combination of the columns of Q. Therefore DORTHO computes the condition number of the matrix $\tilde{Q} := [Q, w/\|w\|] \in \mathbf{R}^{m \times (n+1)}$, where the norm $\| \ \|$ is the Euclidean norm. Using $Q^T Q = I$, we obtain the following expressions for the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{n+1}$ of $\tilde{Q}$:

$$\sigma_1 = (1 + \|Q^T w\|/\|w\|)^{1/2}, \tag{2.1a}$$

$$\sigma_j = 1, \qquad 2 \leq j \leq n, \tag{2.1b}$$

$$\sigma_{n+1} = (1 - \|Q^T w\|/\|w\|)^{1/2}. \tag{2.1c}$$

Further, for $v := (I - QQ^T)w/\|w\|$,

$$\|v\| = \sigma_1 \sigma_{n+1}. \tag{2.2}$$

Since $1 \leq \sigma_1 \leq \sqrt{2}$, σ_{n+1} is also an accurate estimate of the length of the orthogonal projection of $w/\|w\|$ into the null-space of Q^T. In order to avoid severe cancellation of significant digits in (2.1c) we determine first σ_1 from (2.1a) and then σ_{n+1} from (2.2).

Subroutines DINSC and DORTHO have an input parameter RCOND which is a lower bound for the reciprocal condition number. The computations are discontinued and an error flag is set if RCOND $< \sigma_{n+1}/\sigma_1$. On exit, RCOND: $= \sigma_{n+1}/\sigma_1$.

Assume now that the input value of RCOND $\geq \sigma_{n+1}/\sigma_1$. Then DORTHO computes $s := Q^T w$ and $v := (I-QQ^T)w$ by a scheme analogous to the method described by Parlett [15, p. 107] for orthogonalizing a vector against another vector. For definiteness, we present the orthogonalization scheme. References to σ_1, σ_{n+1}, and RCOND are neglected for simplicity.

Orthogonalization algorithm: input $Q \in \mathbf{R}^{m\times n}$ (Q has orthonormal columns), m,n ($m > n$), $w \in \mathbf{R}^m$ ($w \neq 0$); output v ($v = (I-QQ^T)w$), s ($s = Q^T w$);

```
w̃: = w/‖w‖;
s: = Q^T w̃; v: = w̃-Qs;                                   (2.3)
if ‖v‖ ≥ 0.707 then
    v: = v/‖v‖; s: = s‖w‖; exit; * ‖v‖ = 1, Q^T v = 0 *
s′: = Q^T v; v′: = v-Qs′;                                 (2.4)
if ‖v′‖ ≤ 0.707‖v‖ then
    * w lies in span{Q} numerically *
    v: = 0; s: = (s+s′)‖w‖; set flag; exit;
v: = (v+v′)/‖v+v′‖; s: = (s+s′)‖w‖; exit; * ‖v‖ = 1, Q^T v = 0 *
```

The proof in Parlett [15, pp. 107-108] that one reorthogonalization suffices carries over to the present algorithm, using that $Q^TQ = I$.

We note that there are other ways to carry out the computations on lines (2.3)-(2.4). In [5], v and v' are updated immediately after a component of s is computed. Our scheme has the advantages of being faster on vector computers, since it allows matrix vector operations, and it is also, generally, more accurate, since rounding errors

accumulate less. The latter can easily be shown, and we omit the details.

We turn to subroutine DORTHX. This is a faster version of subroutine DORTHO. DORTHX assumes that w in the orthogonalization algorithm is an axis vector. This simplifies the computations in (2.3). DORTHX may perform nearly twice as fast as DORTHO.

2.2 Subroutines DINVIT, DTRLSL and DTRUSL

Given a nonsingular upper triangular matrix $U = [\mu_{jk}] \in \mathbf{R}^{nxn}$ and a vector $b = [\beta_j] \in \mathbf{R}^n$, DTRUSL solves $Ux = b\rho$, where $|\rho| \leq 1$ is a scaling factor such that $|\beta_j\rho/\mu_{jj}| \leq 1$ for all j. The scaling factor is introduced in order to avoid overflow when solving very ill-conditioned linear systems of equations. DTRLSL is an analogous subroutine for lower triangular systems.

DTRLSL and DTRUSL are called by DINVIT, a subroutine for computing an approximation of a right singular vector belonging to a least singular value of a right triangular matrix R. If R is singular then such a singular vector is computed by solving a triangular linear system of equations. Otherwise an initial approximate right singular vector $a^{(o)} = \{\alpha_j^{(o)}\}_{j=1}^n$ is obtained by the LINPACK condition number estimator DTRCO, and inverse iteration with R^TR is used to obtain improved approximations $a^{(j)}$, j = 1,2,...,NMBIT, where NMBIT is an input parameter to DINVIT and DRRPM. On exit from DINVIT and DRRPM, IPOS(j) contains the least index k such that $|\alpha_k^{(j)}| \geq |\alpha_\ell^{(j)}|$, $1 \leq \ell \leq n$, $0 \leq j \leq$ NMBIT. On return from DINVIT and DRRPM the parameter DELTA is given by DELTA: $= \|R^TRa^{(NMBIT)}\|/\|a^{(NMBIT)}\|$. Hence, DELTA is an upper bound for the least singular value of R.

2.3 Updating subroutines

We are in a position to consider the subroutines of Table 1.1. The vectorization is mainly done in the BLAS of the next section, but some loops of the subroutines of Table 1.1 vectorize as well. Comments in the source code reveal which loops vectorize or are eligible for vectorization on an IBM 3090-200VF computer with compiler VS FORTRAN 2.3.0 to where the default vectorization directives are used. For applications to particular problem classes, changing the default vectorization by compiler directives may decrease the execution time.

We list the differences between the subroutines of Table 1.1 and the corresponding Algol procedures of [5]. Some of these modifications were suggested in [5] but not implemented in the Algol procedures [5]. In subroutine DDELC, the column deleted in A: = QR is determined optionally. Not computing this column saves O(mn) arithmetic operations. In subroutine DDELR, the auxiliary subroutine DORTHX is used instead of DORTHO. As indicated in Section 2.1 the former subroutine may perform nearly twice as fast. In subroutine DINSC, a column w is inserted into A: = QR only if the reciprocal condition number of the matrix [Q,w/||w||] is larger than a bound given by the parameter RCOND on entry. The parameter RCOND can be used to prevent updating when w/||w|| is nearly in the range of Q. Finally, DRNK1 performs slightly faster if the updated matrix $A + vu^T$ is such that v lies numerically in the range of A.

The subroutine DRRPM implements an algorithm presented by Chan [3]. The computation of an approximate right singular vector corresponding to a least singular value is done by subroutine DINVIT

and has already been discussed. The position of a component of largest magnitude of this singular vector has to be determined, and we found, in agreement with Chan's suggestion [3], that two inverse iterations suffice. In fact, in all computed examples, one inverse iteration was sufficient, even for problems with multiple or close least singular values. The subroutine permutes the order of columns 1 through k of $A\Pi$ where k is an input parameter, $A \in \mathbf{R}^{m \times n}$, $m \geq n$, and Π is a permutation matrix. DRRPM is typically called with $k = n, n-1, n-2, \ldots$ until no further permutation is made or until the computed upper bound DELTA for the least singular value of the matrix consisting of the first k columns of $A\Pi$ is not small.

The subroutines of Table 1.1 do neither require nor produce a factorization with nonnegative diagonal elements of the upper triangular matrix.

3. THE BLAS

Much computational experience on a variety of computers led Dongarra and Sorensen [7] to conclude that nearly optimal performance of numerical linear algebra subroutines can be achieved if the subroutines for the basic matrix and vector operations, such as multiplication, addition and inner product computation, are written to perform well on vector computers. We wanted to write a code that performs well on an IBM 3090-200VF computer, and that would not require excessive tuning when moved to other (vector) computers. Therefore we designed the code to vectorize well without special compiler instructions, since the latter would be machine dependent.

A feature of the VS FORTRAN 2.3.0 compiler is that unnecessary vector loads and stores are avoided by introducing a temporary scalar variable, denoted by ACC in the subroutine DAPX in Example 3.1. During execution ACC should be thought of as a vector variable stored in a vector register. Timings for DAPX and comparison with code with explicitly unrolled loops have been carried out by Robert and Squazzero [16]. These timings show subroutine DAPX to perform better than equivalent subroutines with explicitly unrolled loops.

Example 3.1. Subroutine for matrix vector multiplication.

```
      SUBROUTINE DAPX(A,LDA,M,N,X,Y)
C
C     DAPX COMPUTES Y:=A*X.
C
      INTEGER LDA,M,N,I,J
      REAL*8 A(LDA,N),X(N),Y(M),ACC
C
C     OUTER LOOP VECTORIZES.
C
      DO 10 I=1,M
        ACC=0D0
        DO 20 J=1,N
          ACC=ACC+A(I,J)*X(J)
20      CONTINUE
        Y(I)=ACC
10    CONTINUE
      RETURN
      END
```

□

Temporary scalar variables have also been used in others of the 17 BLAS used.

4. COMPUTED EXAMPLES

Example 4.1. In this example the QR decomposition of a 4 x 3 matrix A is updated. The use of all subroutines of Table 1.1 is illustrated. The main program producing this output is listed in the Appendix.

```
A=Q*R, MATRIX A:
  0.500   0.000  -0.500
  0.500   0.000  -1.500
  0.500  -1.000   0.500
  0.500  -1.000  -0.500

MATRIX Q:
  0.500   0.500   0.500
  0.500   0.500  -0.500
  0.500  -0.500   0.500
  0.500  -0.500  -0.500

MATRIX R:
  1.000  -1.000  -1.000
  0.000   1.000  -1.000
  0.000   0.000   1.000


DELETE COLUMN 2 OF A AND UPDATE Q AND R BY DDELC
ON RETURN FROM DDELC INFO=0
COLUMN RECOMPUTED BY DDELC:   0.000   0.000  -1.000  -1.000

UPDATED MATRIX Q:
  0.500   0.000
  0.500   0.707
  0.500  -0.707
  0.500   0.000

UPDATED MATRIX R:
  1.000  -1.000
  0.000  -1.414

ABS( ELEMENT OF LARGEST MAGNITUDE OF A-Q*R ):  0.2E-15
ABS( ELEMENT OF LARGEST MAGNITUDE OF Q'*Q-I ): 0.2E-15


DELETE ROW 3 OF A AND UPDATE Q AND R BY DDELR
ON RETURN FROM DDELR INFO=0
ROW RECOMPUTED BY DDELR:    0.500   0.500

UPDATED MATRIX Q:
 -0.577   0.408
 -0.577  -0.816
 -0.577   0.408

UPDATED MATRIX R:
 -0.866   1.443
  0.000   0.816

ABS( ELEMENT OF LARGEST MAGNITUDE OF A-Q*R ):  0.2E-15
ABS( ELEMENT OF LARGEST MAGNITUDE OF Q'*Q-I ): 0.3E-16

NEW 1ST COLUMN TO BE INSERTED INTO A BY DINSC:   1.000   2.000   3.000
ON RETURN FROM DINSC INFO=0
COMPUTED RECIPROCAL CONDITION NUMBER BY DINSC: 0.2E+00

UPDATED MATRIX Q:
 -0.267   0.873   0.408
 -0.535   0.218  -0.816
 -0.802  -0.436   0.408

UPDATED MATRIX R:
 -3.742  -0.802   1.336
  0.000   0.327  -0.546
  0.000   0.000   0.816

ABS( ELEMENT OF LARGEST MAGNITUDE OF A-Q*R ):  0.4E-15
ABS( ELEMENT OF LARGEST MAGNITUDE OF Q'*Q-I ): 0.2E-15
```



```
NEW 3RD ROW TO BE INSERTED INTO A BY DINSR:   1.000   2.000   3.000
ON RETURN FROM DINSR INFO=0

UPDATED MATRIX Q:
 -0.258   0.095  -0.470
 -0.516  -0.095  -0.671
 -0.258   0.949   0.146
 -0.775  -0.285   0.555

UPDATED MATRIX R:
 -3.873  -1.291   0.516
  0.000   1.756   3.085
  0.000   0.000   1.403

ABS( ELEMENT OF LARGEST MAGNITUDE OF A-Q*R ):  0.7E-15
ABS( ELEMENT OF LARGEST MAGNITUDE OF Q'*Q-I ): 0.7E-15


RANK ONE MATRIX V*U' ADDED TO A, Q AND R UPDATED BY DRNK1

VECTOR V:   0.500   0.500   2.000   0.500
VECTOR U:   1.000  -1.000   1.000

ON RETURN FROM DRNK1 INFO=1

UPDATED MATRIX Q:
 -0.275   0.460  -0.097
 -0.458   0.598  -0.398
 -0.550   0.037   0.828
 -0.642  -0.656  -0.384

UPDATED MATRIX R:
 -5.454   0.000  -2.292
  0.000   0.000  -0.412
  0.000   0.000   4.536

ABS( ELEMENT OF LARGEST MAGNITUDE OF A-Q*R ):  0.5E-15
ABS( ELEMENT OF LARGEST MAGNITUDE OF Q'*Q-I ): 0.4E-15


RANK REVEALING QR FACTORIZATION BY DRRPH
ON RETURN FROM DRRPH INFO=0
DELTA ON EXIT FROM DRRPH: 0.8E-16
POSITION OF ELEMENTS OF MAX MAGNITUDE OF SUCCESSIVELY COMPUTED
SINGULAR VECTORS BY DRRPH:  2  2  2
DRRPH DETERMINED QR FACTORIZATION OF MATRIX A(*,IP(J)), WHERE
 IP(1), IP(2), ... =  1  3  2

MATRIX Q FOR COLUMN PERMUTED MATRIX A:
 -0.275   0.138  -0.449
 -0.458   0.450  -0.559
 -0.550  -0.821  -0.112
 -0.642   0.323   0.688

MATRIX R FOR COLUMN PERMUTED MATRIX A:
 -5.454  -2.292   0.000
  0.000  -4.555   0.000
  0.000   0.000   0.000

ABS( ELEMENT OF LARGEST MAGNITUDE OF A-Q*R ):  0.5E-15
ABS( ELEMENT OF LARGEST MAGNITUDE OF Q'*Q-I ): 0.4E-15


MATRIX A AFTER COLUMN PERMUTATION:
  1.500   0.000   0.000
  2.500  -1.000   0.000
  3.000   5.000   0.000
  3.500   0.000   0.000
```

Example 4.2. Execution times for subroutines DDELCO and DRNK1 are compared for scalar and vector arithmetic. The measured cpu times differed somewhat between different executions of the same code. Therefore the reported times are rounded to one significant digit and the quotient of measured cpu times are rounded to the nearest multiple of 1/2.

Table 4.1 shows the cpu times for DDELCO. This routine and its subroutines have been compiled with the VS FORTRAN 2.3.0 compiler. The times for vector arithmetic are obtained from code generated with compiler option vlev = 2, which makes the compiler generate code that utilizes the vector registers and arithmetic. The times for scalar arithmetic are obtained from code generated with compiler option vlev = 0, which makes the compiler generate code that does not use vector instructions. Given a QR decomposition of a matrix $A \in \mathbf{R}^{m \times n}$, Table 4.1 shows the cpu time required by DDELCO to compute the QR decomposition of $\bar{A} \in \mathbf{R}^{m \times (n-1)}$ obtained by deleting column one of A.

		cpu time in seconds		
m	n	scalar arithmetic	vector arithmetic	scalar time / vector time
10	10	$4 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	1
20	10	$4 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	1
30	10	$5 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	1.5
50	10	$6 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	1.5
75	10	$8 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	2
128	10	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	2
1024	10	$7 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	3.5
1280	10	$9 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	3.5

Table 4.1: Timings for DDELCO

Table 4.2 is similar to Table 4.1 and contains execution times for DRNK1. The reduction in execution time obtained by using vector instructions is of the same order of magnitude for the other updating routines, too.

		cpu time in seconds		
m	n	scalar arithmetic	vector arithmetic	scalar time / vector time
16	12	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	1
32	25	$4 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	1.5
64	50	$2 \cdot 10^{-2}$	$7 \cdot 10^{-3}$	2
128	100	$6 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	2.5
1024	100	$4 \cdot 10^{-1}$	$8 \cdot 10^{-2}$	4.5
1250	100	$5 \cdot 10^{-1}$	$9 \cdot 10^{-1}$	5

Table 4.2: Timings for DRNK1

□

Example 4.3. Execution times for subroutines written by Buckley [2] and those of Table 1.1 are compared. The vectorized and scalar codes were generated as explained in Example 4.2. We found that vectorization of the subroutines in [2] did not change the execution times significantly, generally less than 20%. In all computed examples the vectorized subroutines in [2] required at least twice as much execution time than the vectorized subroutines of Table 1.1. For certain problems our vectorized code executed up to 95 times faster than the vectorized code in [2]. For scalar code the differences in execution time often decreased with increasing matrix size. Tables 4 3-4.6 present some sample timings.

m	time for DDELR	time for DELROW [2]	$\frac{\text{time for DELROW [2]}}{\text{time for DDELR}}$
10	$3 \cdot 10^{-5}$	$9 \cdot 10^{-4}$	$3.5 \cdot 10^{1}$
64	$7 \cdot 10^{-5}$	$2 \cdot 10^{-3}$	$2 \cdot 10^{1}$
128	$8 \cdot 10^{-4}$	$3 \cdot 10^{-3}$	3
1024	$4 \cdot 10^{-3}$	$2 \cdot 10^{-2}$	4

Table 4.3: The first row of A = QR is deleted. Cpu times for vectorized code for updating Q and R are given in seconds; n = 10.

m	n	time for DELC	time for DELCOL [2]	$\frac{\text{time for DELCOL [2]}}{\text{time for DDELC}}$
1024	10	$1 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	2
1024	100	$1 \cdot 10^{-4}$	$7 \cdot 10^{-3}$	$7.5 \cdot 10^{1}$
1280	100	$1 \cdot 10^{-4}$	$9 \cdot 10^{-3}$	$9.5 \cdot 10^{1}$

Table 4.4: The last column of A = QR is deleted. Cpu times for vectorized code for updating Q and R are given in seconds. DDELC does not compute the last column of A, i.e., IFLAG = 0 on entry.

m	time for DINSC	time for INSCOL [2]	$\frac{\text{time for INSCOL [2]}}{\text{time for DINSC}}$
64	$1 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	2
128	$1 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	2
1024	$7 \cdot 10^{-3}$	$2 \cdot 10^{-2}$	2.5

Table 4.5: A new first column is inserted into A = QR. Cpu times for vectorized code for updating Q and R are given in seconds; n = 10.

Tables 4.3-4.5 present timings for vectorized code. The next table shows timings for scalar code for the same updatings as in Table 4.3. Table 4.6 shows that, without vectorization, DELROW [2] requires 50% more cpu time than DDELR for moderately large problems.

m	time for DDELR	time for DELROW [2]	time for DELRPW [2] / time for DDELR
10	$2 \cdot 10^{-5}$	$6 \cdot 10^{-4}$	$3 \cdot 10^{1}$
64	$1 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	1.5
128	$2 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	1.5
1024	$1 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	1.5

Table 4.6: The first row of A = QR is deleted. Cpu times for scalar code for updating Q and R are given in seconds; n = 10.

□

ACKNOWLEDGEMENTS

One of the authors (L.R.) would like to thank Pat Gaffney for valuable discussions, and Aladin Kamel for help with handling of the computer.

REFERENCES

[1] Allen, D.M. Mean square error of prediction as a criterion for selecting variables. Technometrics 13 (1971), 469-475.

[2] Buckley, A. Algorithm 580, QRUP: A set of FORTRAN routines for updating QR factorizations. ACM Trans. Math. Software 7 (1981), 548-549 and 8 (1982), 405.

[3] Chan, T.F. Rank revealing QR factorizations. Lin. Alg. Appl. 88/89 (1987), 67-82.

[4] Cline, A.K., Moler, C.B., Stewart, G.W. and Wilkinson, J.H. An estimate for the condition number of a matrix. SIAM J. Numer. Anal. 16 (1979), 368-375.

[5] Daniel, J.W., Gragg, W.B., Kaufman, L. and Stewart, G.W. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. Math. Comput. 30 (1976), 772-795.

[6] Dongarra, J.J., Bunch, J.R., Moler, C.B. and Stewart, G.W. Linpack Users' Guide. SIAM, Philadelphia, 1979.

[7] Dongarra, J.J. and Sorenson, D.C. Linear algebra on high performance computers. In Applications of Supercomputers. Lockhart, D.F., and Hicks, D.L., Eds. Elsevier, Amsterdam, 1986, pp. 57-88.

[8] Gill, P.E., Golub, G.H., Murray, W. and Saunders, M.A. Methods for modifying matrix factorizations. Math. Comput. 28 (1974), 505-535.

[9] Goldfarb, D. Factorized variable metric methods for unconstrained optimization. Math. Comput. 30 (1976), 796-811.

[10] Golub, G.H., and Van Loan, C.F. Matrix Computations. Johns Hopkins Univ. Press, Baltimore, 1983.

[11] Gragg, W.B. and Stewart, G.W. A stable variant of the secant method for solving nonlinear equations. SIAM J. Numer. Anal. 13 (1976), 889-903.

[12] Ling, F., Manolakis, D., and Proakis, J.G. A recursive modified Gram-Schmidt algorithm for least-squares estimation. IEEE Trans. Acoustics, Speech and Signal Processing ASSP-34 (1986), 829-835.

[13] More', J.J., and Sorensen, D.C. Newton methods. In Studies in Numerical Analysis, Golub, G.H., Ed. Math. Assoc. Amer., 1984, pp. 29-82.

[14] NAG FORTRAN Library Manual, Mark 12. Numerical Algorithms Group, 1987.

[15] Parlett, B.N. The Symmetric Eigenvalue Problem. Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[16] Robert, Y., and Sguazzero, P. The LU decomposition algorithm and its efficient FORTRAN implementation on the IBM 3090 vector multiprocessor. Technical Report ICE-0006, IBM European Center for Scientific and Engineering Computing, Rome, 1987.

[17] Stewart, G.W. The effect of rounding error on an algorithm for downdating a Cholesky factorization. J. Inst. Math. Applics. 23 (1979), 203-213.

APPENDIX, Driver for Example 4.1

```
C DRIVER FOR UPDATING THE QR DECOMPOSITION OF A 4 BY 3 MATRIX.
C
      PARAMETER(LDA=4,LDR=3)
      INTEGER M,N,I,K,INFO,J,NMBIT
      INTEGER IPOS(0:5),IP(LDR)
      REAL*8 A(LDA,LDR),Q(LDA,LDR),R(LDR,LDR),B(LDA,LDR)
      REAL*8 WORK(2*LDA+2*LDR+1),W(200),V(LDA),U(LDR)
      REAL*8 RCOND,DELTA
C
C     INITIALIZATION OF A,Q,R,M,N
C
      DATA Q/5D-1,5D-1,5D-1,5D-1,5D-1,5D-1,-5D-1,-5D-1,5D-1,-5D-1,
     %5D-1,-5D-1/
      DATA R/1D0,0D0,0D0,-1D0,1D0,0D0,-1D0,-1D0,1D0/
      DATA A/5D-1,5D-1,5D-1,5D-1,0D0,0D0,-1D0,-1D0,-5D-1,-1.5D0,
     %5D-1,-5D-1/
      DATA M/4/,N/3/
C
      WRITE(6,10)'A=Q*R, MATRIX A:'
10    FORMAT(/1X,70A)
      CALL MATPRI(A,LDA,M,N)
      WRITE(6,*)'MATRIX Q:'
      CALL MATPRI(Q,LDA,M,N)
      WRITE(6,*)'MATRIX R:'
      CALL MATPRI(R,LDR,N,N)
C
C     DELETE COLUMN 2 OF A, UPDATE Q AND R
C
      K=2
      DO 20 I=K+1,N
        DO 30 J=1,M
          A(J,I-1)=A(J,I)
30      CONTINUE
20    CONTINUE
C
      INFO=1
      WRITE(6,10)'DELETE COLUMN 2 OF A AND UPDATE Q AND R BY DDELC'
      CALL DDELC(Q,LDA,M,N,R,LDR,K,V,INFO)
      WRITE(6,40)'DDELC',INFO
40    FORMAT(' ON RETURN FROM ',A5,' INFO=',I1)
C
C     FOR ALL SUBROUTINES INFO=0 ON EXIT INDICATES SUCCESFUL TERMINATION
C
      WRITE(6,50)(V(I),I=1,M)
50    FORMAT(' COLUMN RECOMPUTED BY DDELC:',4F8.3)
      N=N-1
      WRITE(6,10)'UPDATED MATRIX Q:'
      CALL MATPRI(Q,LDA,M,N)
      WRITE(6,*)'UPDATED MATRIX R: '
      CALL MATPRI(R,LDR,N,N)
C
C     PRINT MAGNITUDE OF ELEMENT OF RESIDUAL MATRIX A-Q*R OF LARGEST
C     MAGNITUDE, AS WELL AS FOR MATRIX Q'*Q-I
C
      CALL MAXRES(A,LDA,Q,R,LDR,M,N)
      CALL ORTCHK(Q,LDA,M,N,WORK)
C
C     DELETE ROW 3 OF A, UPDATE Q AND R
C
      K=3
      DO 100 I=1,N
        DO 110 J=K+1,M
          A(J-1,I)=A(J,I)
110     CONTINUE
        A(M,I)=0D0
100   CONTINUE
C
      WRITE(6,10)'DELETE ROW 3 OF A AND UPDATE Q AND R BY DDELR'
      CALL DDELR(Q,LDA,M,N,R,LDR,K,U,WORK,INFO)
      WRITE(6,40)'DDELR',INFO
      WRITE(6,120)(U(I),I=1,N)
120   FORMAT(' ROW RECOMPUTED BY DDELR: ',4F8.3)
      M=M-1
      WRITE(6,10)'UPDATED MATRIX Q:'
      CALL MATPRI(Q,LDA,M,N)
      WRITE(6,*)'UPDATED MATRIX R:'
      CALL MATPRI(R,LDR,N,N)
C
C     PRINT MAGNITUDE OF ELEMENT OF RESIDUAL MATRIX A-Q*R OF LARGEST
C     MAGNITUDE, AS WELL AS FOR MATRIX Q'*Q-I
C
      CALL MAXRES(A,LDA,Q,R,LDR,M,N)
      CALL ORTCHK(Q,LDA,M,N,WORK)
C
C     INSERT NEW FIRST COLUMN V INTO A, UPDATE Q AND R
      K=1
      DO 200 I=N,K,-1
        DO 210 J=1,M
          A(J,I+1)=A(J,I)
210     CONTINUE
200   CONTINUE
      DO 220 I=1,M
        V(I)=I
        A(I,K)=V(I)
220   CONTINUE
      N=N+1
C
C     UPPER BOUND FOR RECIPROCAL CONDITION NUMBER
C
      RCOND=1D-6
C
      WRITE(6,230)(V(I),I=1,M)
230   FORMAT(/1X,'NEW 1ST COLUMN TO BE INSERTED INTO A BY DINSC:',4F8.3)
      CALL DINSC(Q,LDA,M,N,R,LDR,K,V,RCOND,WORK,INFO)
      WRITE(6,40)'DINSC',INFO
      WRITE(6,240)RCOND
240   FORMAT(' COMPUTED RECIPROCAL CONDITION NUMBER BY DINSC:',E8.1)
      WRITE(6,10)'UPDATED MATRIX Q:'
      CALL MATPRI(Q,LDA,M,N)
      WRITE(6,*)'UPDATED MATRIX R:'
```

```
      CALL MATPRI(R,LDR,N,N)
C
C     PRINT MAGNITUDE OF ELEMENT OF RESIDUAL MATRIX A-Q*R OF LARGEST
C     MAGNITUDE, AS WELL AS FOR MATRIX Q'*Q-I
C
      CALL MAXRES(A,LDA,Q,R,LDR,M,N)
      CALL ORTCHK(Q,LDA,M,N,WORK)
C
C     INSERT ROW BETWEEN ROWS 2 AND 3 OF A, UPDATE Q AND R
C
      K=3
      DO 300 J=1,N
        DO 310 I=M,K,-1
          A(I+1,J)=A(I,J)
310     CONTINUE
        U(J)=J
        A(K,J)=U(J)
300   CONTINUE
C
      M=M+1
      WRITE(6,320)(U(I),I=1,N)
320   FORMAT(/1X,'NEW 3RD ROW TO BE INSERTED INTO A BY DINSR:',4F8.3)
      CALL DINSR(Q,LDA,M,N,R,LDR,K,U,WORK,INFO)
      WRITE(6,40)'DINSR',INFO
      WRITE(6,10)'UPDATED MATRIX Q:'
      CALL MATPRI(Q,LDA,M,N)
      WRITE(6,*)'UPDATED MATRIX R:'
      CALL MATPRI(R,LDR,N,N)
C
C     PRINT MAGNITUDE OF ELEMENT OF RESIDUAL MATRIX A-Q*R OF LARGEST
C     MAGNITUDE, AS WELL AS FOR MATRIX Q'*Q-I
C
      CALL MAXRES(A,LDA,Q,R,LDR,M,N)
      CALL ORTCHK(Q,LDA,M,N,WORK)
C
C     ADD RANK ONE MATRIX U*V' TO A, UPDATE Q AND R
C
      DO 400 I=1,M
        V(I)=5D-1
400   CONTINUE
      V(3)=2D0
      DO 410 I=1,N
        U(I)=(-1)**(I+1)
410   CONTINUE
C
      DO 420 J=1,N
        DO 430 I=1,M
          A(I,J)=A(I,J)+U(J)*V(I)
430     CONTINUE
420   CONTINUE
C
      WRITE(6,10)
     %'RANK ONE MATRIX V*U'' ADDED TO A, Q AND R UPDATED BY DRNK1'
      WRITE(6,440)'V',(V(I),I=1,M)
440   FORMAT(/1X,'VECTOR ',1A,':',4F8.3)
      WRITE(6,450)'U',(U(I),I=1,N)
450   FORMAT(' VECTOR ',1A,':',4F8.3)
      CALL DRNK1(Q,LDA,M,N,R,LDR,U,V,WORK,INFO)
      WRITE(6,460)'DRNK1',INFO
C
C     INFO=1 ON EXIT INDICATES THAT V ON ENTRY WAS IN SPAN(A)
C
460   FORMAT(/1X,'ON RETURN FROM ',A5,' INFO=',I1)

      WRITE(6,10)'UPDATED MATRIX Q:'
      CALL MATPRI(Q,LDA,M,N)
      WRITE(6,*)'UPDATED MATRIX R:'
      CALL MATPRI(R,LDR,N,N)
C
C     PRINT MAGNITUDE OF ELEMENT OF RESIDUAL MATRIX A-Q*R OF LARGEST
C     MAGNITUDE, AS WELL AS FOR MATRIX Q'*Q-I. PRINT A.
C
      CALL MAXRES(A,LDA,Q,R,LDR,M,N)
      CALL ORTCHK(Q,LDA,M,N,WORK)
C
C     RANK REVEALING QR FACTORIZATION, INITIALIZE PERMUTATION VECTOR
C
      DO 500 J=1,N
        IP(J)=J
500   CONTINUE
      K=N
      NMBIT=2
      DELTA=10
C
      WRITE(6,10)'RANK REVEALING QR FACTORIZATION BY DRRPM'
      CALL DRRPM(Q,LDA,M,N,R,LDR,K,IP,DELTA,NMBIT,IPOS,WORK,INFO)
      WRITE(6,40)'DRRPM',INFO
      WRITE(6,510)DELTA
510   FORMAT(' DELTA ON EXIT FROM DRRPM:',E8.1)
      WRITE(6,520)(IPOS(J),J=0,NMBIT)
520   FORMAT(' POSITION OF ELEMENTS OF MAX MAGNITUDE OF SUCCESSIVELY ',
     %'COMPUTED',/,' SINGULAR VECTORS BY DRRPM:',513)
      WRITE(6,530)(IP(J),J=1,N)
530   FORMAT(' DRRPM DETERMINED QR FACTORIZATION OF MATRIX ',
     %'A(*,IP(J)), WHERE',/1X,' IP(1), IP(2), ... = ',1013)
      WRITE(6,10)'MATRIX Q FOR COLUMN PERMUTED MATRIX A:'
      CALL MATPRI(Q,LDA,M,N)
      WRITE(6,*)'MATRIX R FOR COLUMN PERMUTED MATRIX A:'
      CALL MATPRI(R,LDR,N,N)
C
C     PERMUTE COLUMNS OF A ACCORDING TO IP, STORE IN B
C
      DO 540 J=1,N
        DO 550 K=1,M
          B(K,J)=A(K,IP(J))
550     CONTINUE
540   CONTINUE
C
C     PRINT MAGNITUDE OF ELEMENT OF RESIDUAL MATRIX B-Q*R OF LARGEST
C     MAGNITUDE, AS WELL AS FOR MATRIX Q'*Q-I
```

```
C
      CALL MAXRES(B,LDA,Q,R,LDR,M,N)
      CALL ORTCHK(Q,LDA,M,N,WORK)
C
      WRITE(6,10)'MATRIX A AFTER COLUMN PERMUTATION:'
      CALL MATPRI(B,LDA,M,N)
C
      STOP
      END
C-----------------------------------------------------------------------
C SUBROUTINES FOR COMPUTING RESIDUALS AND OUTPUT
C-----------------------------------------------------------------------
      SUBROUTINE MATPRI(A,LDA,M,N)
C
C     MATPRI PRINTS MATRIX A
C
      INTEGER LDA,M,N,IROW,ICOL
      REAL*8 A(LDA,N)
C
      DO 10 IROW=1,M
        WRITE(6,20)(A(IROW,ICOL), ICOL=1,N)
10    CONTINUE
      WRITE(6,*)
      RETURN
20    FORMAT(4F8.3)
      END
C--------  ------------------------------------------------------
      SUBROUTINE ORTCHK(Q,LDQ,M,N,WK)
C
C     ORTCHK COMPUTES Q'*Q-I FOR THE M BY N MATRIX Q.
C
      INTEGER LDQ,M,N,IROW,ICOL,K
      REAL*8 Q(LDQ,N),SUM,MX
C
      MX=0D0
      DO 10 IROW=1,N
        DO 20 ICOL=1,N
          SUM=0D0
          DO 30 K=1,M
            SUM=SUM+Q(K,IROW)*Q(K,ICOL)
30        CONTINUE
          IF(IROW.EQ.ICOL)SUM=SUM-1D0
          IF(MX.LT.DABS(SUM))MX=DABS(SUM)
20      CONTINUE
10    CONTINUE
C
      WRITE(6,40)MX
40    FORMAT(' ABS( ELEMENT OF LARGEST MAGNITUDE OF Q''*Q-I ):',
     %E8.1,/1X)
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE MAXRES(A,LDA,Q,R,LDR,M,N)
C
      REAL*8 A(LDA,N),Q(LDA,M),R(LDR,N),SUM,MX
```

```
      MX=0D0
      DO 10 I=1,M
        DO 20 K=1,N
          SUM=A(I,K)
          DO 30 J=1,K
            SUM=SUM-Q(I,J)*R(J,K)
30        CONTINUE
          IF(DABS(SUM).GT.MX)MX=DABS(SUM)
20      CONTINUE
10    CONTINUE
      WRITE(6,40)MX
      RETURN
40    FORMAT(' ABS( ELEMENT OF LARGEST MAGNITUDE OF A-Q*R ):',E9.1)
      END
```

DISTRIBUTION LIST

DIRECTOR (2)
DEFENSE TECH. INFORMATION
CENTER, CAMERON STATION
ALEXANDRIA, VA 22314

DIRECTOR OF RESEARCH ADMIN.
CODE 012
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

LIBRARY (2)
CODE 0142
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

DEPARTMENT OF MATHEMATICS
CODE 53
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

CENTER FOR NAVAL ANALYSES
4401 FORD AVENUE
ALEXANDRIA, VA 22302-0268

PROFESSOR WILLIAM GRAGG (15)
CODE 53Gr
DEPARTMENT OF MATHEMATICS
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

NATIONAL SCIENCE FOUNDATION
WASHINGTON, D.C. 20550